VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Absolvování individuální odborné praxe

Individual Professional Practice in the Company

2018 Luboš Olszar

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Luboš Olszar**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Absolvování individuální odborné praxe
Individual Professional Practice in the Company

Jazyk vypracování: čeština

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: Vikipid a.s.
2. Struktura závěrečné zprávy:
   a) Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta.
   b) Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti.
   c) Zvolený postup řešení zadaných úkolů.
   d) Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe.
   e) Znalosti či dovednosti scházející studentovi v průběhu odborné praxe.
   f) Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení.

Seznam doporučené odborné literatury:

Podle pokynů konzultanta, který vede odbornou praxi studenta.

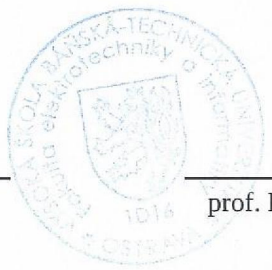
Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Jan Janoušek**

Konzultant bakalářské práce: Bc. Michal Sloviak

Datum zadání: 01.09.2017

Datum odevzdání: 30.04.2018



doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 18. dubna 2018 .....

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.

V Ostravě 18. dubna 2018

Rád bych poděkoval panu Ing. Janu Janouškovi za odbornou pomoc při vypracování této práce. Dále děkuji zaměstnancům firmy VIKIPID a.s. za vřelý přístup, jmenovitě pak Bc. Michalu Sloviakovi za předané zkušenosti. Závěrem chci také poděkovat rodině za podporu při studiu.

Abstrakt

Tato bakalářská práce popisuje mé absolvování odborné praxe ve firmě VIKIPID a.s.. První část práce obsahuje stručné informace o firmě VIKIPID a nastiňuje mé pracovní zařazení. Následující část představuje zadané úkoly, kterými jsem se na praxi zabýval a poté popisuje postup jejich řešení včetně implementace. V závěrečné části jsou shrnuty znalosti a dovednosti, jenž jsem uplatnil či získal, a také celkový přínos odborné praxe.

Klíčová slova: VIKIPID a.s., odborná praxe, ASP.NET, C#, MVC

Abstract

This bachelor thesis describes my professional practice in the VIKIPID a.s. company. The first part of the thesis contains brief information about VIKIPID and about my own work position. The following section introduces tasks I have worked on during the practice, and then describes my suggested solutions, including implementation. The final part summarizes knowledge and skills I have applied or acquired, as well as the overall contribution of professional practice.

Key Words: VIKIPID a.s., professional practice, ASP.NET, C#, MVC

Obsah

Seznam použitých zkratek a symbolů

ASP.NET	–	Active Server Pages .NET
MVC	–	Model View Controller
UML	–	Unified Modeling Language
SQL	–	Structured Query Language
eAPI	–	extended Application Programming Interface
HTML	–	HyperText Markup Language
AES	–	Advanced Encryption Standard
PBKDF2	–	Password-Based Key Derivation Function 2
HMAC	–	Keyed-hash Message Authentication Code
SHA-1	–	Secure Hash Algorithm 1
AJAX	–	Asynchronous JavaScript and XML
XML	–	eXtensible Markup Language
HTTP	–	Hypertext Transfer Protocol
HTTPS	–	Hypertext Transfer Protocol Secure
JSON	–	JavaScript Object Notation
SOAP	–	Simple Object Access Protocol
WSDL	–	Web Services Description Language
PDF	–	Portable Document Format
CSV	–	Comma-Separated Values
MIME	–	Multipurpose Internet Mail Extensions
URL	–	Uniform Resource Locator
URI	–	Uniform Resource Identifier
CSS	–	Cascading Style Sheets
TFS	–	Team Foundation Server

Seznam obrázků

Seznam tabulek

Seznam výpisů zdrojového kódu

1 Úvod

V rámci bakalářské práce absolvovat odbornou praxi ve firmě je zcela jistě výborná příležitost k získání cenných praktických zkušeností. Právě proto jsem si tuto možnost zvolil, abych se dokázal lépe uplatnit po dokončení studia. Dalším důvodem byla i skutečnost, že jsem od února 2017 začal pracovat na částečný úvazek ve společnosti VIKIPID a.s. jako vývojář a nabízelo se mi tak spojit práci se studiem.

Druhou kapitolou této práce bych rád představil firmu VIKIPID a nastínil mé pracovní zařazení v kolektivu této společnosti. Ve třetí kapitole je k nalezení seznámení s úkoly, kterými jsem se v průběhu praxe zabýval. Tyto úkoly jsou však pouze výběrem těch zajímavých ze všech mnou realizovaných. Následující čtvrtá kapitola je pak věnována zvolení postupu a řešení jednotlivých úkolů. Závěrečné kapitoly poukazují na mé uplatněné znalosti ze studia a shrnují zkušenosti, o které mě absolvování odborné praxe obohatilo.

2 Profil společnosti a má pracovní pozice

2.1 Profil společnosti

VIKIPID a.s. je společnost, jež byla založena v Ostravě roku 2012. Její vizí bylo zajistit zákazníkům větší bezpečí ve světě online nakupování a zároveň zvýšit obchodníkům efektivitu prodeje. Firma vyvinula pro Českou republiku systém VIKIPID, který obsahuje řadu služeb. Systém je propojen se systémy vybraných dopravců, díky čemuž jsou zásilky realizované skrz VIKIPID nepřetržitě v dohledu. Společnost je tak schopna proplácet obchodníkovi hodnotu dobírky v den, kdy zásilku předají dopravci. Naproti tomu garantuje zákazníkovi vrácení peněz v případě nedoručení zásilky. VIKIPID patří mezi hrstku českých subjektů, které získaly licenci platební instituce od České národní banky.[1] Součástí služeb systému je také garanční platební brána, se kterou firma obsadila třetí místo v soutěži *Inovační firma Moravskoslezského kraje 2017*. Aktuálně je již ve vývoji systém pro evropský trh.

Systém VIKIPID je vyvíjen v jazyce C# na platformě .NET, přičemž veškeré webové aplikace sloužící zákazníkům, či interní správě, jsou postaveny na technologii ASP.NET MVC.

2.2 Pracovní zaměření

Po nástupu do společnosti jsem se stal součástí tříčlenného týmu programátorů. Mým úkolem bylo zapájet se do vývoje ve všech směrech od společné analýzy problémů přes návrh řešení až po samotnou implementaci. Při své práci jsem tak musel vytvářet či upravovat SQL struktury systému, implementovat změny do systémových aplikací, vyvíjet webové aplikace a ojediněle také tvořit UML diagramy aktivit pro přehlednější testování částí systému. Pro mou znalost Bootstrap knihovny jsem mimo rozsáhlejší úkoly také často upravoval front-end aplikací.

3 Zadání úkolů

3.1 Tvorba webové aplikace VIKIPID Account

Nově vyvíjený systém pro evropský trh potřeboval webovou aplikaci pro klienty společnosti VIKIPID, kde by obchodník viděl zejména přehled objednávek, faktury, aktualizace a mohl vytvářet akce jako refundovat platbu, vyjádřit se k reklamacím či si aktivovat služby. Mým úkolem tedy bylo takovou multijazyčnou aplikaci pomocí technologie ASP.NET MVC vytvořit. Tahle práce byla nejrozsáhlejší ze všech, které jsem v průběhu absolvování odborné praxe řešil.

3.2 Napojení na systém dopravce DPD

VIKIPID systém pro Českou republiku obsahuje napojení na systémy vybraných dopravců. Mým úkolem bylo zajistit napojení na systém dopravce DPD, čili vytvořit knihovnu, která bude schopná komunikovat se systémem DPD pomocí jejich poskytované služby. Dle dodané dokumentace jsem měl zajistit funkce jako vytváření objednávek, objednávání svozu na zásilky, získávání informací o zásilkách a další. Součástí úkolu bylo i vytvořit generování korektních přepravních štítků, které se lepí na zásilky a obsahují všechny informace nutné ke správné evidenci zásilek v průběhu doručování.

3.3 Implementace platební brány v režimu iframe

Systém VIKIPID je propojen s eAPI platební brány ČSOB. Zákazník přistupuje k platební bráně skrz webovou aplikaci VIKIPID Brána, ve které si před přesměrováním k platbě může přidat k nákupu doplňkovou službu VIKIPIDu. Mou úlohou bylo přepracovat právě onu webovou aplikaci na zcela odlišnou, jež bude obsahovat novou službu. Zároveň celý průběh platby se měl na e-shopech nově zobrazovat v režimu iframe.

3.4 Časová náročnost úloh

Úloha	Počet dní
Tvorba webové aplikace VIKIPID Account	40
Napojení na systém dopravce DPD	14
Implementace platební brány v režimu iframe	16

Tabulka 1: Časová náročnost úloh

4 Řešení zadaných úkolů

4.1 Tvorba webové aplikace VIKIPID Account

Systém VIKIPID přijímá objednávky ze systémů e-shopů, u kterých pak nadále sleduje průběh stavů a pracuje s němi. Aby měli obchodníci využívající systém VIKIPID přístup k informacím objednávek, bylo potřeba vytvořit webovou aplikaci VIKIPID Account. V rámci zadání jsem od vedení dostal čtyři obrázky, jež znázorňovaly jak by přibližně mělo vypadat uživatelské rozhraní základních podstránek aplikace. Konkrétně se jednalo o přehled transakcí k objednávkám, zprávy, přehled služeb a fakturace. Finální aplikace však obsahuje mnohem více podstránek, které jsem mohl stylovat dle vlastního uvážení. Rozšíření aplikace o nové funkce jsem ovšem vždy konzultoval s vedením. Dalším požadavkem bylo zajistit multijazyčnost, aby mohl VIKIPID Account sloužit pro evropský trh. Tvorba této aplikace mi zabrala zhruba 40 pracovních dní, ověřil jsem si však naplno své schopnosti a dovednosti v kompletním vývoji webových aplikací.

4.1.1 Popis aplikace a její řešení

VIKIPID Account je vyvíjen jako všechny ostatní webové aplikace společnosti v ASP.NET MVC frameworku. Úvodem bych se chtěl zmínit, v čem spočívá princip MVC frameworku a MVC softwarové architektury. MVC je zkratkou pro Model, View a Controller, což jsou na sobě nezávislé komponenty, které společně tvoří celkovou aplikaci. Tyto komponenty reprezentují doménový model, uživatelské rozhraní a doménovou logiku, přičemž jsou vzájemně odděleny, aby i při častých změnách během vývoje bylo možné zajistit flexibilitu a spolehlivost aplikace.[2] Objekty modelu jsou části aplikace, které implementují logiku datové domény aplikace a obsahují například i výpočty validace. Model je prakticky klasická C# třída. View zobrazuje výstup uživateli a vytváří tak uživatelské rozhraní, které je převážně vytvářeno z modelu. View je zobrazováno jazykem HTML, ovšem v ASP.NET lze používat Razor syntaxi, která zjednodušuje zápis kódu. Výhodou je, že je možné ve view používat také některé C# konstrukce, jako například proměnné, podmínky, cykly, aj. Controller je takovým prostředníkem mezi modelem a view. Obstarává interakce uživatele, plní objekty modelů daty, odpovídá uživateli a vybírá view, které se bude do uřivatelského rozhraní renderovat.[3] Pro napojení aplikace na databázi systému je použit Entity framework, který nesmírně usnadňuje práci.

Při vývoji jsem jako první řešil přihlášení do aplikace. Uživatelské jméno i heslo jsou citlivé data, a proto jsou šifrovány. Výhodou je, že .NET framework nabízí možnost využít jmenný prostor System.Security.Cryptography, jenž obsahuje škálu tříd a metod pro práci se šifrováním. K šifrování i dešifrování údajů jsem použil symetrický algoritmus AES, který šifruje i dešifruje stejným klíčem. Pro získání tohoto klíče jsem využil třídu Rfc2898DeriveBytes, která implementuje klíčové derivační funkce PBKDF2 užívající pseudonáhodný generátor čísel na základě HMAC-SHA-1.[4] U PBKDF2 se k šifrování mimo master password přidává i tzv. salt byte, jenž se také zohledňuje při výpočtu finálního klíče a zvyšuje tak zabezpečení údajů.

```
public string Encrypt(string clearText)
{
   byte[] clearBytes = Encoding.Unicode.GetBytes(clearText);
   using (Aes encryptor = Aes.Create())
    {
      Rfc2898DeriveBytes pdb = new Rfc2898DeriveBytes(SecretKey, new byte[] { 0
          x39, 0x76, ... });
        encryptor.Key = pdb.GetBytes(32);
        encryptor.IV = pdb.GetBytes(16);
        using (MemoryStream ms = new MemoryStream())
        {
         using (CryptoStream cs = new CryptoStream(ms, encryptor.CreateEncryptor
             (), CryptoStreamMode.Write))
            {
               cs.Write(clearBytes, 0, clearBytes.Length);
                cs.Close();
            }
            clearText = Convert.ToBase64String(ms.ToArray());
        }
    }
    return clearText;
}
```

Výpis 1: Funkce pro šifrování údajů (salt byte je zkrácený)

Při přihlašování uživatele do aplikace se vytváří hash string uživatelského jména i hesla a postupně se předávají do funkce Encrypt (Výpis 1), která vrací konečné zašifrované údaje. Tyto šifry se pak hledají v příslušné tabulce systému a v případě shody může být uživatel přihlášen, v opačném případě je upozorněn na nesprávně zadané údaje. Controller ještě naplní z databáze globální objekt User, který obsahuje informace o uživateli, jako jsou ID, jméno, ID firmy pod kterou je uživatel evidován, kód nastaveného jazyka, aj. Tyto údaje přihlášeného uživatele poté slouží napříč aplikací. Aby měly controllery přístup k objektu User, dědí všechny ze základního AppController, kde se objekt nachází. Na základě uloženého jazyka uživatele se nastaví CurrentCulture a CurrentUICulture vlákna aplikace, v případě prvního přihlášení je nastaven výchozí anglický jazyk. Při přihlášení se také uživateli do prohlížeče ukládá autentizační cookie.

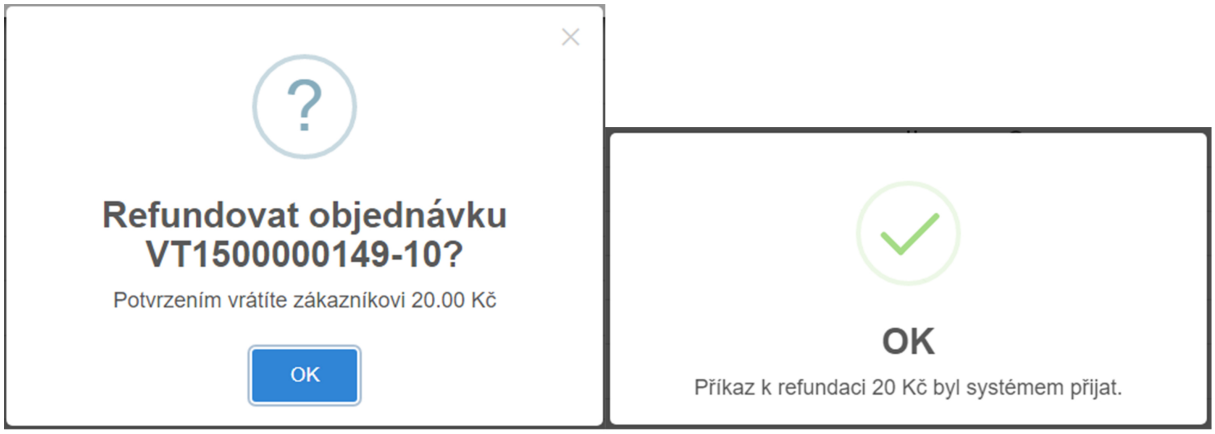
Obrázek 1: VIKIPID Account - přehled transakcí

Hlavní stránkou aplikace jsou tzv. transakce (Obrázek 1), které ukazují přehled transakcí k objednávkám. V této tabulce vidí obchodník to nejpodstatnější. Konkrétně zda byla objednávka zákazníkem zaplacena, v jakém se nachází stavu, zda VIKIPID systém již proplatil obchodníkovi částku za objednávku, termín nejzazšího doručení a další. Navíc obsahuje funkční ikony pro otevření detailu objednávky a stornování objednávky. Pokud byla objednávka zaplacena a obchodník již nevrátil zákazníkovi plnou částku, objeví se tlačítko pro refundaci (vrácení částky zákazníkovi). Lze refundovat část celkové ceny za zásilku i plnou částku. Po stisknutí tlačítka se v modálním okně zobrazí formulář, kde uživatel vyplní hodnotu částky, kterou si přeje refundovat. Tohle pole je při každé změně kontrolované JavaScriptem, zda je hodnota korektní a nepřesahuje celkovou částku objednávky. V případě správnosti vstupu je zpřístupněno tlačítko pro vykonání refundace. Zákazník je ještě pro jistotu dotázán dalším informačním modálním oknem, zda si skutečně přeje refundovat objednávku o danou částku (Obrázek 2). Při potvrzení se pomocí technologie AJAX zasílá HTTP POST request na metodu controlleru TransactionPayback. V této metodě dojde znovu ke kontrole vstupu, aktualizuje se cena objednávky, založí se záznam do tabulky s historií cen objednávky a zároveň se uloží do příslušné tabulky systému záznam o zadané refundaci. S touto tabulkou posléze pracuje Finanční servis VIKIPIDu, který zadává příkazy do systému Multicash k provedení finančních transakcí. Aby bylo možné uživateli zobrazit výsledek požadavku, metoda TransactionPayback vrací jako HTTP response objekt ve formátu JSON. Tento objekt (Výpis 2) obsahuje bool hodnotu, zda proběhl příkaz k refundaci úspěšně či nikoliv a zároveň obsahuje samotnou zprávu v aktivním jazyce aplikace. JavaScript tak na základě objektu v odpovědi dokáže zobrazit uživateli správné okno sdělující úspěch či chybu.

```
public class ResponseModel
{
   public bool Success { get; set; }
   public string ResponseText { get; set; }

   public void Set(bool success, string resposneText)
   {
      Success = success;
      ResponseText = resposneText;
   }
}
```

Výpis 2: Třída sloužící k odpovědím AJAX HTTP requestů

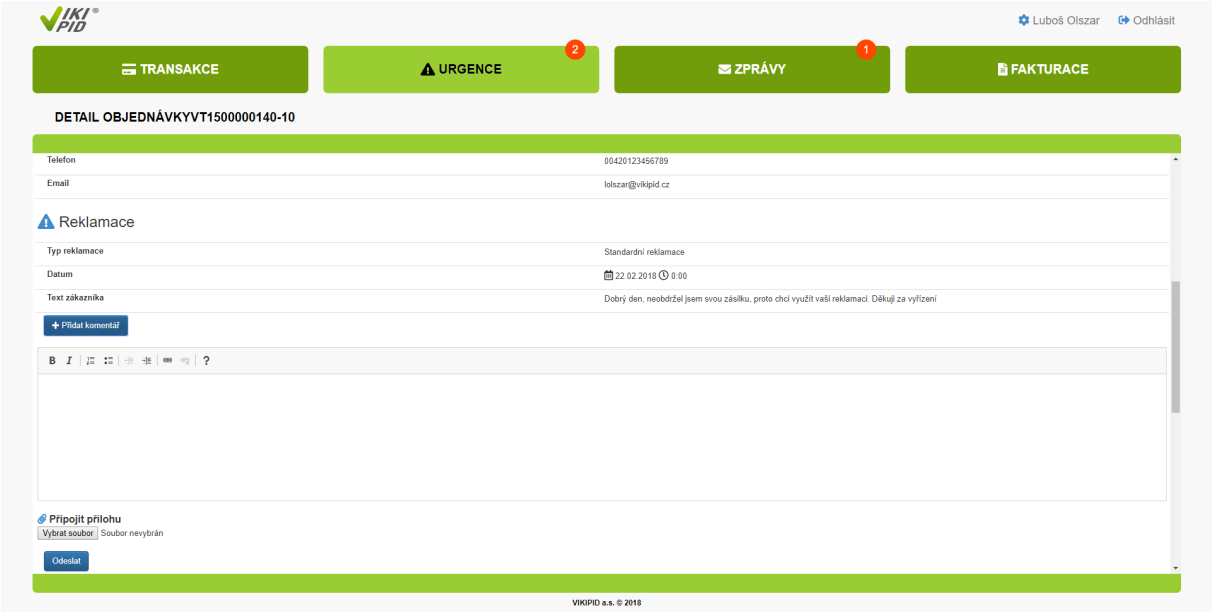


Obrázek 2: VIKIPID Account - modální okna v průběhu refundace

Další stránkou ve hlavním menu jsou urgence, jenž jsou v podstatě filtrované transakce, které se nacházejí ve stavu reklamace. Tyto reklamace je obchodník povinen řešit, jinak služba systému VIKIPID v určitém limitu automaticky uzná reklamaci za oprávněnou. Právě proto mají tyto objednávky samostatnou sekci v menu, doplněnou o informativní ikonu udávající počet nevyřízených reklamací. Obchodník se může vyjádřit k reklamaci pomocí formuláře v detailu objednávky (Obrázek 3), který tvoří textový editor a text box pro nahrání přílohy. K vytvoření textového editoru jsem použil open source plugin CKEditor.

Jelikož HTTP POST request tohoto formuláře obsahoval soubor, zvýšil jsem v konfiguraci webové aplikace (Web.config) maximální možnou velikost souboru na 10 MB. Tohle nastavení skrývá atribut maxRequestLenght v elementu <httpRuntime>. Při zpracovávání formuláře dochází v controlleru ke kontrole přípony souboru. Nenachází-li se jeho přípona v definovaném poli povolených přípon, akce je zamítnuta. Pokud je typ souboru povolen, dojde k jeho unikátnímu přejmenování, uloží se na straně serveru do vyhrazené složky a založí se v databázi VIKIPID

systému nový záznam o přidání komentáře k reklamaci. Tyto komentáře poté řeší a vyhodnocují zaměstnanci společnosti VIKIPID skrz interní webovou aplikaci pro správu systému.



Obrázek 3: VIKIPID Account - přidání komentáře k reklamaci

Zprávy slouží v aplikaci VIKIPID Account jako taková jednosměrná schránka pro přijímání zpráv od VIKIPIDu. Tato část byla na implementaci poměrně jednoduchá, poprvé jsem se však setkal s tím, že bych musel zabraňovat rodičům HTML elementu spouštění eventů. Jelikož na řádek v tabulce zpráv je vázán event onClick (otevře zprávu v modálním okně) a zároveň ve sloupci řádku se nachází ikona, která pomocí Url.Action volá metodu controlleru (stažení přílohy), docházelo k tomu, že při kliknutí na ikonu stažení se také otevírala zpráva. To je způsobeno tím, že spouštění eventů v HTML DOM stromě objektů probublává směrem nahoru ke kořeni. Zabránil jsem tomu však poměrně jednoduše pomocí jQuery metody event.stopPropagation().

Poslední stránkou v hlavním menu jsou fakturace, kde obchodník pochopitelně vidí přehled všech vystavených faktur. Společnost VIKIPID používá k účetnictví software Microsoft Dynamics NAV, který vytváří faktury a uchovává jejich data. K získání těchto dat poskytuje Web services, který má komunikaci založenou na SOAP protokolu (uplatňuje také HTTP protokol) a přenášené zprávy jsou standardně ve formátu XML. Fakturace ve VIKIPID Account tak komunikují s touto službou a získávají informace, jako zda již byla faktura zaplacená, kolik zbývá k úhradě, datum splatnosti, aj. Uživatel má ke každému záznamu v tabulce k dispozici ke stažení dva soubory. Prvním je samotná faktura ve formátu PDF, druhý soubor je datový ve formátu CSV a slouží systémům e-shopů k parsování dat. Ke stažení těchto souborů jsou využívány obdobné metody, lišící se pouze cestou k souboru na serveru a MIME typem souboru (application/pdf, application/msexcel).

```
public ActionResult DownloadInvoice(string filename)
{
   try
   {
      string path = $"{Db.Settings.First(o => o.ValueName == "InvoiceFolder").
         Value}\\{User.CompanyId}\\{User.EShopId}\\{filename}";
      FileStream fs = new FileStream(path, FileMode.Open, FileAccess.Read);
      return File(fs, "application/pdf", filename);
   }
   catch (Exception e)
   {
      CreateVikipidException(e, "DownloadInvoice", $"Nepodarilo se stahnout
         fakturu {filename}, eshop ID-{User.EShopId}.", "Invoice");
      AddMessageToTempData(Invoice.DownloadError, true);
      return RedirectToAction("Index", "Invoice");
   }
}
```

Výpis 3: Metoda pro stažení faktury ze serveru

Dalším požadavkem vedení bylo zajistit z důvodů bezpečnosti automatické odhlášení uživatele z aplikace po dvaceti minutách neaktivity, což se dá programátorsky zajistit hned několika způsoby. Jedním z řešení by mohl být JavaScript umístěný v layoutu stránky. Dnes již také vím, že .NET framework nabízí elegantní způsob řešení v konfiguračním souboru Web.config, ovšem v době mé praxe jsem implementoval vlastní způsob v souboru Global.asax. Soubor Global.asax je volitelný ASP.NET soubor, který obsahuje kód pro obsluhu událostí ze strany serveru a odpovídá aplikační vrstvě. V době běhu aplikace je soubor Global.asax oddělen a kompilován do dynamicky generované třídy .NET frameworku odvozené ze základní HttpApplication třídy.[5] Pro mou práci jsem využil metodu, která se spouští pokaždé, když je vyvolán event Application_BeginRequest. Již název napovídá, že tento event je vyvoláván při každém requestu aplikace a právě v těchto okamžicích se mi hodilo kontrolovat, zda-li od posledního spuštění této metody neuběhlo 20 minut. V metodě (Výpis 4) ukládám na straně klienta cookie, jež obsahuje hodnotu aktuálního času navýšeného o 20 minut. Vždy na začátku metody tuhle HttpCookie vytáhnu z kolekce cookies requestu a zkontroluji, jestli je její čas menší než čas aktuální. Pokud ano, přesměruji request na URL odhlášení z aplikace, jinak aktualizuji hodnotu cookie a uložím ji do cookie kolekce objektu response. V aplikaci však existují i stránky, ke kterým je povolen anonymní přístup (např. přihlašovací stránka), a proto je veškerá logika metody obalená v podmínce, která nevpouští hlouběji tyto výjimky. Zároveň jsem chtěl zajistit, aby se uživateli po opětovném přihlášení zobrazila ihned stránka, kde poslední request mířil. To jsem vyřešil

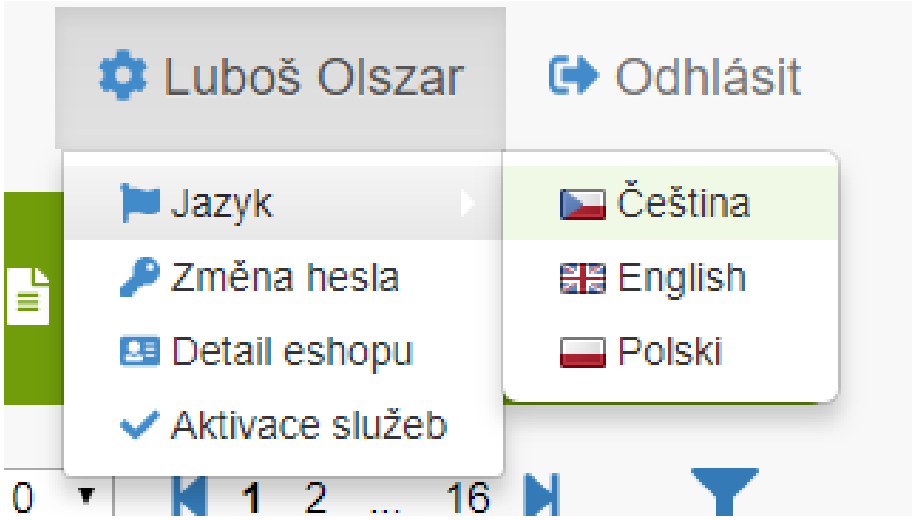
přidáním query stringu, který v případě odhlášení uchovává v URL parametr returnUrl, jehož hodnota je právě relativní cesta posledního requestu. Tu pak metoda controlleru pro přihlášení v případě existence použije k přesměrování na danou stránku.

```
protected void Application_BeginRequest(object sender, EventArgs e)
{
   if (Request.CurrentExecutionFilePath != "/User/LogOff" && Request.
       CurrentExecutionFilePath != "/User/Login" && /*...*/)
   {
      HttpCookie activityCookie = Request.Cookies["u_activity"];

      if (activityCookie != null)
      {
         DateTime date;
         if (DateTime.TryParse(activityCookie.Value, CultureInfo.
             InvariantCulture, DateTimeStyles.None, out date) && date < DateTime
             .UtcNow)
         {
            Response.Redirect("/User/LogOff?activity=0&returnUrl=" + Request.
                CurrentExecutionFilePath.ToString());
         }
         else
         {
            activityCookie.Value = DateTime.UtcNow.AddMinutes(20).ToString(
                CultureInfo.InvariantCulture);
            Response.Cookies.Add(activityCookie);
         }
      }
      else{ /*Vytvoreni activityCookie a redirect na odhlaseni*/ }
   }
    else{ /*Vytvoreni activityCookie*/ }
}
```

Výpis 4: Metoda souboru Global.asax pro automatické odhlášení (kód je zkrácený)

VIKIPID Account skrývá mimo již popisované stránky a funkce ještě v záhlaví aplikace dropdown menu, které obsahuje přístup k veškerému nastavení uživatele (Obrázek 4). Tyto sekce již nebudu detailně popisovat, pouze krátce shrnu jejich podstatu. V první řadě se zde nachází volba jazyka, která mimo změnu jazykové kultury aplikace provádí navíc update uživatele v databázi systému (mění atribut nastaveného jazyka). Další položkou v menu je změna hesla, kde má uživatel možnost si v jednoduchém formuláři změnit heslo. Detail e-shopu slouží obchodníkovi k přehledu veškerých informací, co dodal společnosti VIKIPID při podpisu smlouvy. Zde nemůže uživatel nic editovat, ovšem při jakékoliv změně je povinen kontaktovat VIKIPID. Poslední položkou jsou aktivace služeb sloužící k aktivaci či deaktivaci poskytovaných služeb VIKIPIDu. Zda je nebo není určitá služba aktivní ovlivňuje různé faktory. Příkladem může být třeba v jaké fázi objednávky bude systém VIKIPID zasílat e-shopu peníze nebo zda poskytne obchodník svým zákazníkům nadstandardní službu garance ve VIKIPID platební bráně.



Obrázek 4: VIKIPID Account - dropdown menu v záhlaví aplikace

Závěrem bych chtěl k webové aplikaci VIKIPID Account říci, že jsem se při tvorbě front-endu držel zásad knihovny Bootstrap a i s pomocí media queries v CSS je aplikace responzivní pro různá rozlišení desktopů, laptopů a tabletů. Veškeré vektorové ikony v aplikaci pochází z freeware Font Awesome sady ikon.

4.2 Napojení na systém dopravce DPD

Systém VIKIPID obsahuje škálu knihoven řešící napojení na různé dopravce. Mým úkolem bylo vytvořit takovou knihovnu pro dopravce DPD. Tato knihovna měla obsahovat sadu metod hlavně pro zajištění automatizace zasílání dat dopravci, objednávání svozu na zásilky a získávání stavů zásilek při přepravě. K zadání jsem obdržel tzv. příručku programátora pro integraci zákaznických řešení a DPD systémů pomocí webových služeb MojeDPD. Součástí úkolu bylo i vytvoření přepravního štítku. Implementace celkem rozsáhlé knihovny včetně testování mi zabrala zhruba 14 dní.

4.2.1 Web Services MojeDPD

DPD poskytuje ke kompletní správě zásilek webové služby MojeDPD, jejichž rozhraní je popsáno jazykem WSDL. Web Services komunikují protokolem SOAP (využívá protokol HTTP a jeho metodu POST) a přijímají objekty ve formátu XML.

MojeDPD poskytuje celkem tři služby:

- ShipmentService (výpočet ceny, vytvoření zásilek, tisk přepravních štítků)
- ManifestService (vytváření seznamu zásilek a dokončení expedice zásilek)
- PickupOrderService (objednávání svozu na vytvořené zásilky)

Dokumentace DPD obsahuje popis metod služeb prostřednictvím tabulek, které ukazují všechny vstupní parametry, datový typ, zda-li jsou povinné a jejich stručný popis. Zároveň stejným způsobem popisuje, jaké elementy obsahuje response objekt.

4.2.2 Postup a popis řešení napojení

V prostředí Visual Studia se u projektu nachází možnost pro přidání service reference, což bylo mým prvním krokem při řešení úkolu. Prostřednictvím této možnosti jsem získal z webových služeb knihovny, obsahující jejich metody a definice objektů, což umožňuje právě WSDL popis.

Začal jsem řešením základní metody createShipment pro vytváření zásilek v DPD databázi systému, jejíž implementace zabrala zhruba 400 řádků kódu. To je způsobeno hlavně tím, že za různých podmínek se vyplňují jiné parametry objektu ShipmentVO, jenž reprezentuje zásilku. DPD navíc poskytuje svým zákazníkům spoustu možných doplňkových služeb, jako je například večerní doručení, které je možné využít pouze na území České republiky a ještě jen v určitých městech. Aby bylo možné získat aktuální restrikce a omezení PSČ k produktům DPD, je poskytována další doposud nezmiňovaná webová služba, která je oddělena od Web Services MojeDPD. Pomocí ní jsem byl již schopný vytvořit kupříkladu funkci, jež kontroluje, jestli příjemce s určitým PSČ může využít službu večerního doručení (Výpis 5). Během vytváření struktur objektů ShipmentVO tak dochází k celé škále obdobných kontrol, zda-li je možné zásilku s požadovanými službami vytvořit.

```
private bool IsZipCodeValidForEveningDelivery(string zipCode)
{
   if(!string.IsNullOrEmpty(zipCode))
   {
        Iproduct_apiClient api = new Iproduct_apiClient();
        Restriction[] res = api.GetValidRestrictionsForProduct("841", "");
        int numberZip;
        bool result = Int32.TryParse(zipCode, out numberZip);

        if(result)
        {
            foreach (var item in res)
            {
               if (numberZip >= int.Parse(item.zipcode_from) && numberZip <= int
                   .Parse(item.zipcode_to))
                {
                  return true;
                }
            }
        }
    }
   return false;
}
```

Výpis 5: Funkce pro validaci PSČ ke službě DPD večerní doručení

Volání funkce IsZipCodeValidForEveningDelivery předchází podmínka vpouštějící hlouběji pouze české příjemce. Po vytvoření struktur objektů se může zavolat na službě ShipmentService metoda createShipment, která vrací response objekt obsahující list odpovědí ke všem zásilkám. Tyto odpovědi posléze v cyklu kontroluji, jestli došlo při uložení zásilky do systému DPD k chybě. Pokud ano, založím záznam do error logu VIKIPID databáze s popisem odpovědi, v opačném případě se bude se zásilkou dále pracovat. V následujících krocích ve zkratce dochází ke kontaktování služby metodou, která vrací čárový kód zásilky a následně další, jež poskytuje přepravní kód zásilky. Tyto obě hodnoty se ukládají do systému VIKIPID k objednávce a jsou nezbytné pro budoucí možnost generování štítku a sledování zásilky.

Podle dokumentace jsem postupoval v implementaci dalších metod, které však z důvodu repetitivnosti v implementaci již nebudu detailně popisovat. Jednalo se prakticky vždy o obdobný scénář s tvorbou rozdílných struktur objektů a s využitím rozdílných metod webových služeb. Pro představu však zmíním, že celá knihovna je napsána v přibližně 1700 řádcích.

4.2.3 Generování přepravních štítků

VIKIPID poskytuje v českém systému obchodníkům službu VIKIPID Štítek. Jedná se o službu, která umožňuje vygenerování přepravních štítků zásilek do pdf dokumentu. Obchodník si poté tyto štítky vytiskne a umístí na své zásilky. DPD poskytuje dokumentaci, která obsahuje přesné parametry a rozložení všech nezbytných údajů jejich přepravního štítku. K tomu dodává sadu textových souborů, jejichž obsahem jsou data, jež se mohou objevit na štítku. Konkrétně je to třeba seznam všech evropských dep DPD, seznam možných popisů při kombinaci poskytovaných služeb a další.

K vytváření přepravního štítku jsem použil metody třídy XGraphics z jmenného prostoru PdfSharp.Drawing. PDF dokument, ze kterého se vytváří instance třídy XGraphics, reprezentuje objekt třídy PdfDocument z jmenného prostoru PdfSharp.Pdf. Základním aspektem přepravního štítku je čárový kód, pro jehož renderování do obrazové podoby jsem použil freeware .NET komponentu Spire.Barcode. Celá metoda pro generování přepravních štítků obsahuje podobně jako u napojení na systém DPD řadu podmínek. Například zásilka na dobírku bude obsahovat jiné informace než již zaplacená zásilka a podobně.

Každá transportní služba využita k přepravě zásilky má své jedinečné označení číslem, tohle ID transportní služby má systém VIKIPID u každé objednávky uloženo. Právě jeden z dodaných textových souborů obsahuje jednoduchou „tabulku“ dvou atributů, jež popisuje jaký text se má k jednotlivým transportním službám uvádět. Například, že pro ID s hodnotou 352 se má na štítku zobrazit text „DPD 8:30 dobirka / C.O.D.“. Zde se mi hodilo parsovat data ze souboru do kolekce Hashtable, jejíž výhodou je, že mapuje libovolné klíče k jejich hodnotám. Tahle kolekce zajistila přehledný a jednoduchý přístup k textovým popisům transportních služeb. Zápis pro získání hodnoty je stejný jako u klasického pole, avšak jako index pohodlně uvádím ID transportní služby.

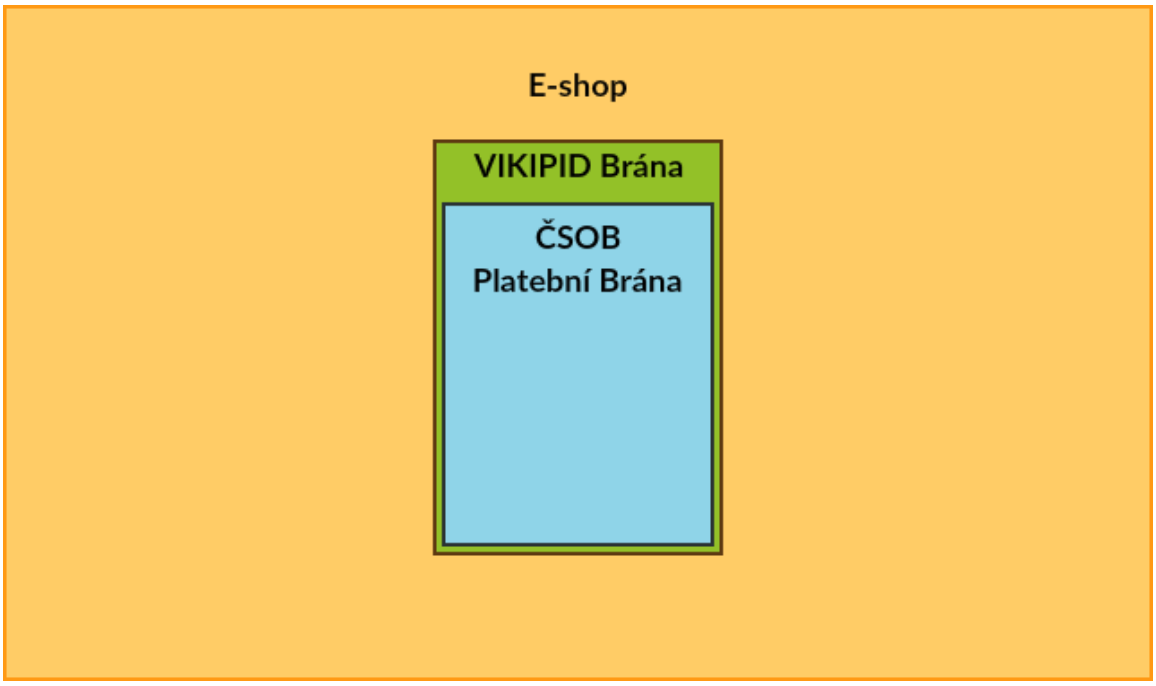
Obrázek 5: Vygenerovaný DPD přepravní štítek

4.3 Implementace platební brány v režimu iframe

Společnost VIKIPID využívá platební bránu od ČSOB, ke které přistupuje skrz vlastní webovou aplikaci VIKIPID Brána. Tato webová aplikace byla již zastaralá a bylo potřeba aktualizovat její prostředí i vzhledem k nově poskytované službě. Vedení požadovalo rovnou kompletní předělání i po vzhledové stránce, implementování zobrazení brány v režimu iframe a přidání multijazyčnosti. Tento režim má přinést nakupujícím pocit většího bezpečí, když na pozadí platby se stále nacházejí na stránce e-shopu. Před mým zpracováním tohoto úkolu byli nakupující ve svém prohlížeči přesměrováni nejdříve na VIKIPID Bránu a poté znovu přesměrováni na stránku ČSOB brány. V režimu iframe by zákazníci neměli postřehnout, že se jim zobrazuje obsah zcela odlišných stránek a celý průběh platby má působit jednotně. ČSOB dodává příručku k implementaci iframe režimu a vedení VIKIPIDu mi dalo obrázkovou předlohu cílové webové aplikace. Řešení úkolu jsem se věnoval přibližně 16 dní.

4.3.1 Iframe zobrazení

Úvodem bych rád vysvětlil, co to vůbec iframe zobrazení je. Jedná se o HTML či XHTML tag, v jehož obsahu je možné zobrazit jinou webovou stránku. Tato vnořená webová stránka nesdílí s vnější CSS styly ani jiné prvky.[6] Zkrátka není možné z vnější stránky přetěžit CSS styly stránky vnitřní. V případě tohoto úkolu se bude nakupujícím zobrazovat dvojitý iframe. Nejprve první, pomocí kterého se na e-shopu zobrazí VIKIPID Brána a ta následně obsahuje v závěrečném kroku platby druhý, jenž vkládá do webové aplikace ČSOB platební bránu.



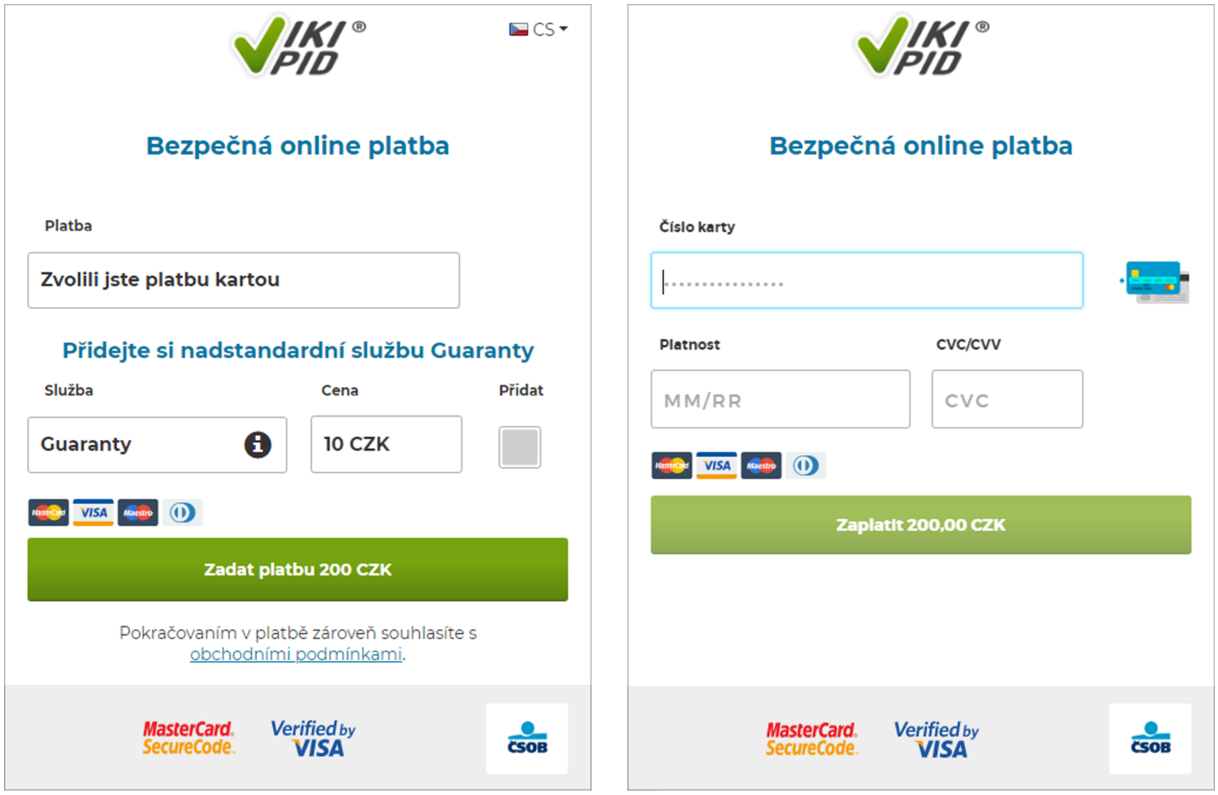
Obrázek 6: Nákres sestavení obsahu stránky v prohlížeči (závěrečný krok platby)

4.3.2 Postup a popis řešení

Napojení na eAPI ČSOB již bylo v minulosti řešeno jiným kolegou, avšak z důvodů mých zásahů jsem se nejprve prostřednictvím online dokumentace seznámil s životními cykly transakcí a s hlavními operacemi eAPI. Nejprve VIKIPID Brána spouští operaci Init, která zakládá platební transakci na straně ČSOB. V případě úspěšného založení může dojít k přesměrování na platební bránu ČSOB, kde zákazník provede platbu. Poté dochází ke zpětné POST odpovědi na VIKIPID Bránu, která obsahuje informaci o stavu platby.

Abych se dostal k popisu řešení VIKIPID Brány, popíši prvotně co předchází jejímu zobrazení. V momentě kdy zákazník na e-shopu dokončí objednávku, je posílán request na vstupní servis VIKIPID systému. Objednávka je zpracována, uložena do databáze VIKIPIDu a následně servis vrací odpověď e-shopu. V odpovědi se mimo jiné nachází URL adresa, která vede k platbě objednávky právě na VIKIPID Bráně. Součástí URL adresy je parametr id, jenž obsahuje hash kód objednávky. Index metoda VIKIPID Brány čte tento parametr z URL pomocí HTTP GET metody a na základě něj renderuje view. Pokud v databázi systému není nalezena žádná objednávka s takovým hashem, zobrazí se chybové hlášení. V případě, že taková objednávka existuje, kontrolují se další aspekty, jako zda již byla zaplacená nebo zda již byla inicializovaná platba na danou objednávku. Zákazník může totiž dojít k platbě i prostřednictvím URL odkazu, jenž obdržel na svůj e-mail, a proto jsou tyto kontroly nezbytné. Za standardních okolností však většina těchto výjimek nenastane a hlavním rozcestníkem je až podmínka, která rozděluje e-shopy s aktivní a neaktivní službou Guaranty (Tuto službu si obchodník zapíná/vypíná ve webové aplikaci VIKIPID Account). Pokud má e-shop tuto službu aktivní, zobrazí se ve VIKIPID Bráně nejprve možnost přidat si tuto službu k platbě. V opačném případě je tento krok přeskočen, dochází rovnou k inicializaci platby operací Init a zobrazí se ČSOB platební brána v iframe okně pod záhlavím VIKIPID Brány.

V prvním kroku se zákazníkovi zobrazí VIKIPID Brána v jazyce, který byl zaslán e-shopem při žádosti vytvoření objednávky v systému. Tento jazyk si však zákazník může v záhlaví aplikace změnit. Cena za službu Guaranty je počítaná funkcí a vždy se tedy liší v závislosti na ceně objednávky a ceníku, jenž má e-shop v systému přidělen. Zákazník si tuto službu může přidat k platbě pomocí checkboxu, který v tlačítku níže změní částku celkové platby prostřednictvím JavaScriptu. Po stisku tlačítka se technologií AJAX zasílá HTTP POST request na Index metodu controlleru, ve které dojde k inicializaci platby a konstrukcí PartialView vrací jako odpověď sestavený HTML obsah ČSOB platební brány v iframe bloku (případně view s chybovou hláškou). HTML blok prvního kroku je poté nahrazen obsahem odpovědi a zákazník tak může zrealizovat platbu.



Obrázek 7: VIKIPID Brána - první a následný druhý krok (obsahuje iframe ČSOB brány) platby

Z bezpečnostních důvodů je povoleno zobrazení VIKIPID Brány v režimu iframe pouze webům užívajícím HTTPS protokol. Proto jsem implementoval v Index metodě controlleru také kontrolu tzv. HTTP refereru. Tento údaj je zapsán v hlavičce HTTP dotazu webovým prohlížečem a označuje URI, ze kterého byla webová stránka navštívena.[7] Pokud tedy obsahem není HTTPS protokol, nebude se renderovat view prvního či druhého kroku platby, nýbrž view s patřičným chybovým hlášením. E-shopy jenž nevlastní web s HTTPS protokolem tedy nemohou implementovat iframe zobrazení, avšak mohou standardně zákazníky přesměrovat na stránku VIKIPID Brány, kde lze platbu zrealizovat v plném zobrazení.

5 Využité znalosti a dovednosti získané ze studia

Studiem na univerzitě jsem získal řadu znalostí, které se mi hodily v průběhu mého absolvování odborné praxe. Z hlediska objektově orientovaného programování jsem nabyl vědomosti již v prvním ročníku v předmětu *Programování II* a v následujícím ročníku mi pomohly *Programovací jazyky I* a *Programovací jazyky II* si lépe osvojit objektově orientované programování. Již tehdy mě více zaujal jazyk C# a platforma .NET Framework, proto jsem si vybral jako volitelný předmět *Architektura technologie .NET*, kde jsem získal pokročilejší znalosti, jenž jsem v průběhu praxe uplatňoval. Zcela určitě mé programové smýšlení obohatily i předměty *Algoritmy I* a *Algoritmy II*. Pro vytváření databázových struktur a SQL dotazů mi naprosto stačila znalost získána z předmětů *Úvod do databázových systémů* a *Databázové a informační systémy*. Předmětem *Úvod do softwarového inženýrství* jsem se naučil UML jazyk, který byl nezbytný při čtení dokumentací či při tvorbě vlastních diagramů sloužících k ulehčení testování. K vytvoření tohoto dokumentu popisující mou odbornou praxi jsem použil LaTeX,jehož znalosti jsem získal z předmětu *Elektronické publikování*.

6 Chybějící znalosti a dovednosti v průběhu praxe

Během absolvování odborné praxe jsem pochopitelně ve svých dovednostech narazil i na řadu nedostatků, které jsem však vždy dokázal samostudiem doplnit. Mezery jsem měl zpočátku hlavně při tvorbě front-endu aplikací. Seznamoval jsem se s knihovnou Boostrap, která se již začíná standardně užívat a doplňoval jsem i celkové znalosti HTML a CSS. V rámci ASP.NET jsem se naučil používat Razor syntaxi, o které jsem sice na univerzitě v předmětu *Architektura technologie .NET* slyšel, avšak ji detailně neznal. Poprvé jsem také pracoval s verzovacím systém TFS od Microsoftu, což je obdobou známějšího systému GIT.

7 Závěr

Před nástupem do společnosti VIKIPID a.s. jsem se obával, do jaké míry budu schopen plnit zadané úkoly. Byl jsem ovšem velice mile překvapen faktem, že univerzita mi skutečně dala široký přehled v IT sféře a plnit úkoly často nebyl žádný problém. Samozřejmě mi pomáhalo samostudium, které je v oblasti programování nezbytné, ale i vřelý přístup kolegů. Seznámil jsem se s vývojem v kolektivu včetně moderních technologií, jenž se užívají.

Zdokonalil jsem své dovednosti nejvyšší mírou převážně v programování webových aplikací, konkrétně v MVC frameworku. Mimo odborné znalosti, o kterých se již zmiňuji v předešlé kapitole, jsem se naučil také zamýšlet se nad problémy, následně navrhovat co jak nejoptimálnější řešení a odhadovat čas implementace řešení úkolů. Těší mě, že jsem nedělal pouze drobné úpravy na systému, ale že mi byla po pár měsících vkládána důvěra a pracoval jsem i na větších projektech.

Můj celkový pocit z absolvování této odborné praxe je pozitivní a myslím si, že mě tato zkušenost velice obohatila. Výhodou je také fakt, že po ukončení studia již budu disponovat nějakou praxí, která je na trhu práce velmi ceněna.

Literatura

[1] O firmě VIKIPID a.s. *VIKIPID a.s.* [online]. [cit. 2018-04-04]. Dostupné z: `http://www.vikipid.cz/#about-us`

[2] Model View Controller. *Wikipedie: Otevřená encyklopedie* [online]. [cit. 2018-03-16]. Dostupné z: `https://cs.wikipedia.org/wiki/Model-view-controller`

[3] ASP.NET MVC Overview. *Microsoft Developer Network* [online]. [cit. 2018-03-16]. Dostupné z: `https://msdn.microsoft.com/en-us/library/dd381412`

[4] Rfc2898DeriveBytes Class. *Microsoft Developer Network* [online]. [cit. 2018-25-03]. Dostupné z: `https://msdn.microsoft.com/en-gb/library/system.security.cryptography.rfc2898derivebytes`

[5] Global.asax Syntax. *Microsoft Developer Network* [online]. [cit. 2018-27-03]. Dostupné z: `https://msdn.microsoft.com/en-us/library/2027ewzw(v=vs.100).aspx`

[6] IFRAME. *Wikipedie: Otevřená encyklopedie* [online]. [cit. 2018-04-10]. Dostupné z: `https://cs.wikipedia.org/wiki/IFRAME`

[7] HTTP referer. *Wikipedie: Otevřená encyklopedie* [online]. [cit. 2018-04-11]. Dostupné z: `https://cs.wikipedia.org/wiki/HTTP_referer`